

## ГЛАВА 5. Специализированная фреймовая экспертная система `iu4.expert.lab`

Экспертная система предназначена для построения баз знаний на основе фреймовых семантических сетей, предназначенных для решения задач синтеза и классификации. Тестовые примеры наглядно поясняют принципы работы экспертных систем данного класса. Этот пакет позволяет:

1. Вести несколько отдельных проектов на одной аппаратной платформе
2. Создавать и редактировать фреймовую семантическую сеть
3. Тестировать экспертную систему, созданную на основе фреймовой семантической сети
4. Выводить отчет о найденных вариантах решения

Интерфейс программы показан на рисунке 5.1., 5.2

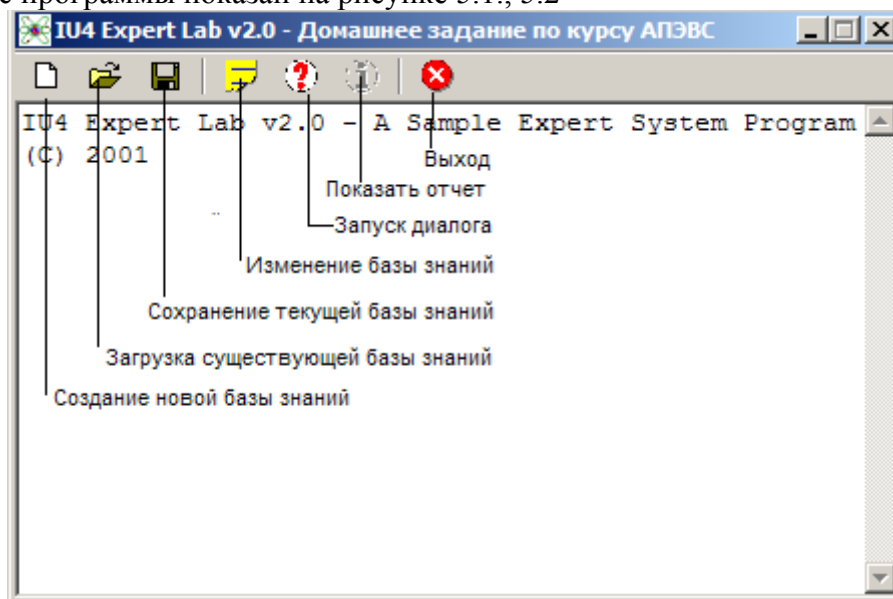


Рис.5.1 Основное окно программы

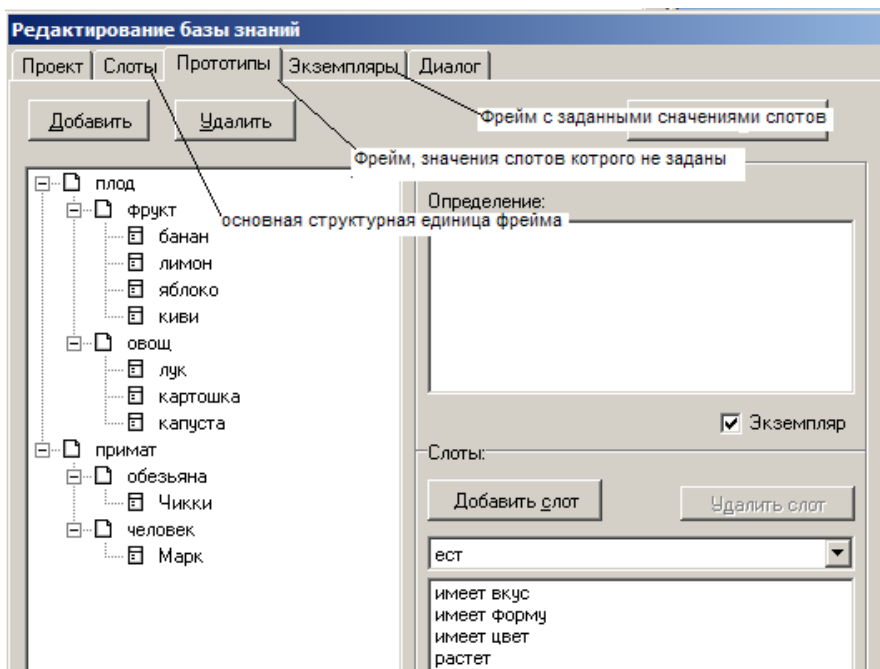


Рис.5.2 Окно редактирования базы знаний.

Первый шаг работы с экспертной системой – работа с машиной ввода, первый шагом необходимо заполнить информационную карту проекту проекта рис. 5.3.

The screenshot shows a window titled "Редактирование базы знаний" (Editing knowledge base) with a tabbed interface. The "Проект" (Project) tab is active. It contains several text input fields: "Название проекта:" (Project name) with the value "Экспертная система интерактивного подбора технологического процесса", "Назначение:" (Purpose) with the value "Подбор технологического процесса сборки электронной ячейки по косвенным признакам.", "Проблемная область:" (Problem area) with the value "Технологические процессы сборки электронных модулей.", and "Автор:" (Author) with the value "Червинский А.С., Панфилова С.П.". On the right side, there are four buttons: "Ok", "Очистить" (Clear), "Загрузить" (Load), and "Сохранить" (Save).

Рис.5.3. Свойства проекта экспертной системы

Далее выявляются объекты предметной области и их свойства. Свойства объектов объединяются в группы, которые определяются слотами, например, виды монтажа, выводы компонентов и т.п. Названия слотов заносятся в справочник слотов.

The screenshot shows the same "Редактирование базы знаний" window, but with the "Слоты" (Slots) tab active. It features a list of slots, with the first one selected and its details displayed in a text area: "Вид монтажа" (Mounting type), "выводы компонентов" (Component pins), "есть микросхемы с планарными выводами" (There are microchips with planar pins), "мощность компонента (количество штрихов)" (Component power (number of strokes)), and "устанавливаемые компоненты" (Installable components). Above the list are "Добавить" (Add) and "Удалить" (Delete) buttons. The right side of the window has the same "Ok", "Очистить", "Загрузить", and "Сохранить" buttons.

Рис.5.4. Слоты экспертной системы интерактивного подбора технологического процесса

Далее формируется дерево фреймовой семантической сети, которое описывает иерархию объектов предметной области и их взаимосвязи (Рис.5.5.)

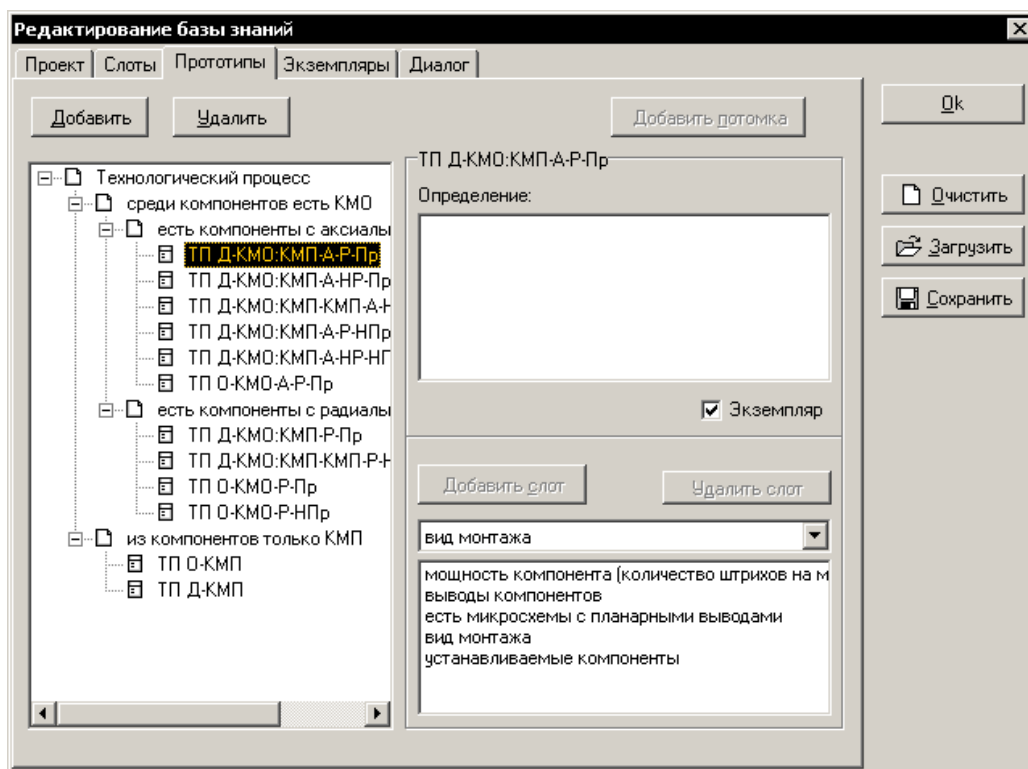


Рис.5.5. Прототипы экспертной системы интерактивного подбора технологического процесса

Исходными данными для разработки фреймовой семантической сети является «дерево узлов» иерархической декомпозиции технологического процесса, полученного в ходе выполнения технологического аудита в модели IDEF0.



Рис.5.6. Дерево узлов модели ТП IDEF0.

В левом окне программы строится фреймовое дерево – фреймов-прототипов, путем задания названия фрейма и установки иерархической взаимосвязи. Для слотов фреймов нижнего уровня декомпозиции назначаются слоты, которые определяют свойства фреймов, для фреймов высшего уровня декомпозиции свойства наследуются снизу вверх. Весь перечень доступных в справочнике слотов доступен в выпадающем списке, для назначения слота фрейму необходимо выбрать его в

списке и нажать на кнопку – добавить слот. В разделе «определение» даются поясняющие комментарии для каждого из фреймов и слотов при необходимости.

На рис.5.7. изображены экземпляры экспертной системы интерактивного подбора технологического процесса. В форме для каждого из фреймов прототипов назначаются конкретные значения и формируются фреймы экземпляры.

Слот	Значение
○ мощность компонента (количес...	больше 2
○ выводы компонентов	аксиальные
○ есть микросхемы с планарными...	есть
○ вид монтажа	двусторонний
○ устанавливаемые компоненты	КМО:КМП

Рис.5.7. Экземпляры экспертной системы интерактивного подбора технологического процесса

Целевой фрейм: Стробоскоп

Установки машины вывода

Приглашение:

Диалог:

Выводить при успешном завершении:

Найдено решение:

Выводить при отсутствии решения:

Решений не найдено:

☐ Автоматически создавать отчет

Рис.5.8. Установка целевого фрейма и редактирование фраз для машины вывода.

После завершения работы с машиной ввода можно переходить к работе с машиной вывода (рис.5.10.) в режиме диалога.

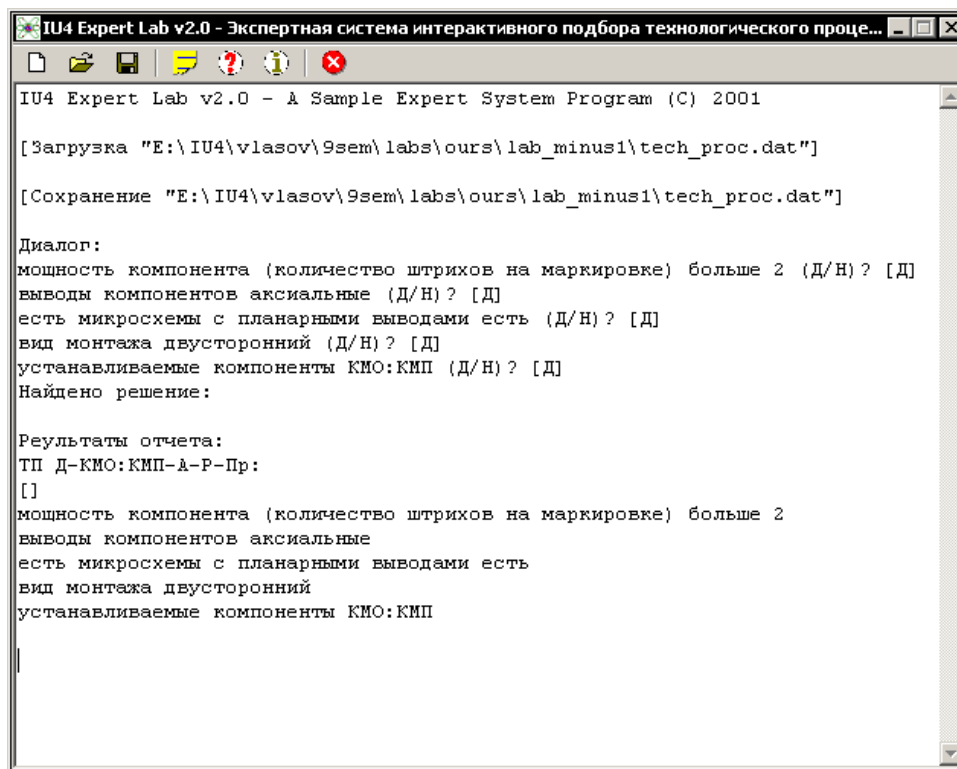


Рис.5.9. Диалог пользователя с машиной вывода полученной ЭС

**Задание:** изучить пример базы знаний, реализованной в файле fruits.dat

Описание предметной области: Объекты: плоды и приматы, Свойства объектов задаются слотами: ест, живет, имеет форму и т.д. прототипов плодов (фрукт, овощ), приматов (обезьяна, человек) и их конкретных экземпляров: банан, картошка, Марк и т.д.

Диалог с экспертной системой состоит в ответах (да,нет) на вопросы вида: <слот><значение слота>, например «живет в Африке (Д/Н)?»

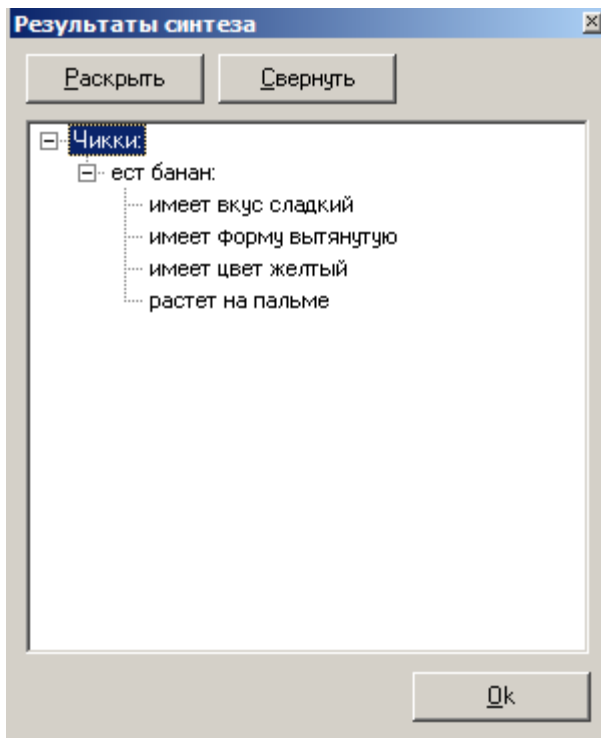


Рис.5.10. Отчет о результатах синтеза.

В результате диалога с пользователем, экспертная система формирует, например, следующее решение:

**Диалог:**

ест фрукт (Д/Н)? [Д]

живет в Африке (Д/Н)? [Д]

Это маленькая обезьяна Чикки //для слота  
«ест» производится декомпозиция  
что же ест Чикки?

имеет вкус сладкий (Д/Н)? [Д]

имеет форму вытянутую (Д/Н)? [Д]

имеет цвет желтый (Д/Н)? [Д]

растет на пальме (Д/Н)? [Д]

Это банан

Таблица 1.1. Листинг программы на Си , реализующей экспертную систему на основе фреймовой семантической сети

```
#include "stdio.h"
#include "ctype.h"
#include "conio.h"
#include "string.h"
#include "malloc.h"
#include <bios.h>
#include <stdlib.h>
#include <dos.h>
#include <mem.h>
#include <dir.h>
#include <io.h>

#include "popup.h"
#include "kode.h"
#include "mouse.h"

#define MAX 100 //Размер массива K_base

struct attribute
{char attrib[80]; struct attribute *next;}at; //Список связанных элементов

struct object{char name[80]; struct attribute *alist;}ob; //Указатель на список
атрибутов
struct object k_base[MAX]; //Массив базы знаний об объектах
struct attribute *yes, *no;
struct attribute *yesnext, *nonext;

int l_pos=-1;

free_trails(void); //освобождение памяти занимаемой элементов БЗ
menu_expert(void); //меню
enter(void); //ввод объекта и списка его атрибутов
query(void); //машина ввода: запрос информации у эксперта.
save(void); //сохранение базы знаний в файле
load(void); //ввод базы знаний из файла
kont_ob(struct attribute *p, char *ob); //проверка подходящего объекта
clear_kbase(void); //восстановление базы знаний
ask(char *attrib); //проверяет, что аттребут уже опрашивался
trailno(struct attribute *p); //проверяет содержание в объекте лишних аттребутов
trailyes(struct attribute *p); //проверяем наличие в объекте необходимых
аттребутов
get_next(void); //получения следующего индекса при доступе к
массиву k_base
is_in(char ch, char*s); //сервисная функция
extern void expert(void); //основная функция обеспечения работы экспертной
системы

windesc *w_menu, *wind, *winq, *wine;

void expert(void)
{
char ch;
no=yes='\0'; //Инициализируем указатели

w_menu=draw_win(CTRWIN, 0, 80, 3,"Меню работы с базой знаний:L-загрузка, D-
диалог, Q-выход из режима", popup, &defcolors);

do
{
free_trails();
```

```
ch=menu_expert();
switch(ch)
{
    case 'e': enter(); break;
    case 'd': query(); break;
    case 's': save(); break;
    case 'l': load(); break;
}
}while(ch!='q');

rmv_win(w_menu);
clrscr();
}

//Функция освобождения памяти занимаемой элементом.
free_trails()
{
    struct attribute *p;

    while(yes)
    {
        p=yes->next;
        free(yes);
        yes=p;
    }
    while(no)
    {
        p=no->next;
        free(no);
        no=p;
    }
    return;
}

//Функция ввода объекта и списка его атрибутов
enter()
{
    windesc *w_error;
    int t;//Индекс следующего свободного элемента в k_base
    int i;
    struct attribute *p, *oldp;

    wine=draw_win(CTRWIN, 4, 80, 16,"Редактирование базы знаний", popup,
&defcolors);

    for(;;)
    {
        t=get_next();//получаем индекс следующего доступного элемента в БЗ
        if(t==-1)
        {
            w_error=draw_win(CTRWIN, CTRWIN, 40, 3, "", popup, &errcolors);
            mprintf("\r\nСписок объектов БЗ исчерпан, для продолжения нажмите любую
клавишу");
            if(!kbhit());
            rmv_win(w_error);clrscr();
            return;
        }
        mprintf("Наименование системы гашения");
        mprintf("\r\n:");
        gets(k_base[t].name);
        mprintf("\r\n ");
        if(!*k_base[t].name)//Введено пустое имя? Да
        {
            l_pos--;
```



```

        break;
    }
    p=(struct attribute*) malloc(sizeof(at));
    if(p=='\0')//выделяем память под список атрибутов
    {
        w_error=draw_win(CTRWIN, CTRWIN, 40, 3, "Ошибка", popup, &errcolors);
        mprintf("Нет доступной памяти");
        if(!kbhit());
        rmv_win(w_error);clrscr();
        return;
    }
    k_base[t].alist=p;//p-адрес начала списка атрибутов
    for(i=0;i<sizeof(p->attrib);i++) p->attrib[i]=' ';
    for(;;)
    {
        mprintf("Характеристика исходного акуст. поля");
        mprintf("\r\n:");
        gets(p->attrib);
        mprintf("\r\n");
        if(!p->attrib[0]) break;
        oldp=p;
        p->next=(struct attribute*)malloc(sizeof(at));

        if(p->next=='\0'){
            w_error=draw_win(CTRWIN, CTRWIN, 40, 3, "Ошибка", popup, &errcolors);
            mprintf("Нет доступной памяти");
            if(!kbhit());
            rmv_win(w_error);clrscr();
            return;
        }
        p=p->next;
        p->next='\0';
        for(i=0; i<sizeof(p->attrib);i++)p->attrib[i]=' ';
    }
    oldp->next='\0';
}
rmv_win(wine);
slct_win(w_menu);
return;
}

//Функция запроса информации у эксперта
query()
{
    int t;
    char ch;
    struct attribute *p;

    winq=draw_win(CTRWIN, 20, 80, 6,"Результаты синтеза", popup, &defcolors);
    wind=draw_win(CTRWIN, 4, 80, 16,"Диалог:Является ли исходное поле:", popup,
&defcolors);

    for(t=0;t<=l_pos;t++)
    {
        p=k_base[t].alist;
        if(kont_ob(p,k_base[t].name))
        {
            slct_win(winq);
            mprintf("Для гашения акустического поля с указанными свойствами
рекомендуется:\r\n%s",k_base[t].name);
            mprintf("\r\n                                     Для продолжения нажмите любую клавишу!");
            goto quit1;
        }
    }
}

```

```
slct_win(winq);
mprintf("В базе знаний нет системы удовлетворяющей предъявленным требованиям");
mprintf("\r\n                          Для продолжения нажмите любую клавишу!");
quit1:

while(!kbhit());
rmv_win(wind);
rmv_win(winq);
slct_win(w_menu);
return;
}

//Функция поиска и проверки подходящего объекта
kont_ob(struct attribute *p, char *ob)
{
    windesc *w_error;
    char answer;
    struct attribute *a, *t;
    if(!trailno(p)) return 0;
    if(!trailyes(p)) return 0;

    slct_win(wind);

    while(p)
    {
        //Проверяем если атрибут уже опрашивался, то переходим к следующему
        if(ask(p->attrib))
        {
            mprintf("%s (Y/N)?", p->attrib);
            answer=tolower(getche());
            mprintf("\r\n");
            a=(struct attribute*) malloc(sizeof(at));
            if(!a)
            {
                w_error=draw_win(CTRWIN, CTRWIN, 35, 3, "Ошибка", popup, &errcolors);
                mprintf("Нет доступной памяти");
                if(!kbhit());
                rmv_win(w_error);clrscr();
                return 0;
            }
            a->next='\0';
            switch(answer)
            {
                case 'n': strcpy(a->attrib, p->attrib);
                    if(!no)
                    { no=a;
                      nonext=no;
                    }
                    else
                    { nonext->next=a;
                      nonext=a;
                    }
                    return 0;
                case 'y': strcpy(a->attrib,p->attrib);
                    if(!yes)
                    {yes=a;
                      yesnext=yes;
                    }
                    else
                    {
                        yesnext->next=a;
                        yesnext=a;
                    }
                    p=p->next;
            }
        }
    }
}
```

```

        break;
    case 'w': //Объяснение выбора сделанного экспертом
        mprintf("Проверяется %s\n",ob);

        if(yes) mprintf("Это средство гашения:\n",ob);
        t=yes;
        while(t){
            printf("%s\n",t->attrib);
            t=t->next;}

        if(no) mprintf("Оно не предназначено для гашения акуст. поля с
характеристикой:\n");
        t=no;
        while(t){
            printf("%s\n",t->attrib);
            t=t->next;}
        break;
    }
    }
    else p=p-> next;
}
return 1;
}

//Функция проверки содержит ли объект атрибуты которые у него должны
отсутствовать
trailno(struct attribute *p)
{
    struct attribute *a, *t;
    a=no;
    while(a){
        t=p;
        while(t){
            if(!strcmp(t->attrib,a->attrib)) return 0;//Объект обладает атрибутом, который
должен отсутствовать
            t=t->next;}
        a=a->next;}
    return 1;
}

//Функция проверки содержит ли объект все те атрибуты, которые он должен
содержать
traiyes(struct attribute *p)
{
    struct attribute *a, *t;
    char ok;
    a=yes;
    while(a)
    {
        ok=0;
        t=p;
        while(t){
            if(!strcmp(t->attrib,a->attrib)) ok=1;//Объект обладает атрибутом
            t=t->next;}
        if(!ok) return 0;
        a=a->next;
    }
    return 1;
}

//Функция опроса, что атрибут уже опрашивался
ask(char *attrib)
{
    struct attribute *p;

```

```
p=yes;
while(p&&strcmp(attrib, p->attrib))
p=p->next;
if(!p) return 1;//Ложь, если достигнут конец списка
else return 0;
}

// Сервисная функция получения следующего доступного индекса в массиве k_base
get_next()
{
    l_pos++;
    if(l_pos<MAX) return l_pos;
    else return -1;
}

//Сервисная функция создания меню
menu_expert()
{
    int i, key;
    char ch;

    slct_win(w_menu);

    do{
        clrscr();
        mprintf("Выберите опцию:");
        ch=tolower(getche());
    } while(! is_in(ch,"eqsld"));
    return ch;
}

//Функция сохранения базы знаний
save()
{
    windesc *w_error;
    int t,x;
    struct attribute *p;
    FILE *fp;

    if((fp=fopen("expert.dat","w"))==0)
    {
        w_error=draw_win(CTRWIN, CTRWIN, 25, 3, "Ошибка", popup, &errcolors);
        mprintf("Невозможно открыть файл");
        delay(1000);
        return;
    }

    w_error=draw_win(CTRWIN, CTRWIN, 25, 3, "Сохранение", popup, &errcolors);
    mprintf("Сохранение базы знаний");

    for(t=0;t<=l_pos;++t)
    {
        for(x=0; x<sizeof(k_base[t].name); x++)
            putc(k_base[t].name[x],fp);
        p=k_base[t].alist;
        while(p)
        {
            for(x=0;x<sizeof(p->attrib);x++)
                putc(p->attrib[x],fp);
            p=p->next;
        }
        for(x=0;x<sizeof(p->attrib);x++)//Достигнут конец списка указателей
            putc('\0',fp);
    }
}
```

```
putc('\0',fp);
fclose(fp);

rmv_win(w_error);clrscr();

return;
}

//Функция загрузки базы знаний
load()
{
    windesc *w_error, *w_error1;
    int t,x;
    struct attribute *p, *oldp;
    FILE *fp;

    if((fp=fopen("expert.dat","r"))==0)
    {
        w_error=draw_win(CTRWIN, CTRWIN, 40, 3, "Ошибка", popup, &errcolors);
        mprintf(" Невозможно открыть файл expert.dat");
        delay(1000);
        return;
    }

    w_error=draw_win(CTRWIN, CTRWIN, 42, 3, "Загрузка Базы Знаний", popup,
    &errcolors);
    mprintf("Одну минуточку, загружается база знаний");
    delay(1000); //задержка для возможности чтения предыдущего сообщения

    //Освобождаем память занимаемую старым списком
    clear_kbase();

    for(t=0;t<MAX;++t)
    {
        if((k_base[t].name[0]=getc(fp))==0) break;
        for(x=1;x<sizeof(k_base[t].name);x++) k_base[t].name[x]=getc(fp);
        k_base[t].alist=(struct attribute*)malloc(sizeof(at));
        p=k_base[t].alist;

        if(!p){
            w_error1=draw_win(CTRWIN, CTRWIN, 45, 3, "Ошибка", popup, &errcolors);
            mprintf("Нет доступной памяти,нажмите любую клавишу");
            while(!kbhit());
            rmv_win(w_error1);clrscr();
            return;
        }
        for(;;)
        {
            for(x=0;x<sizeof(p->attrib);x++)
                p->attrib[x]=getc(fp);
            if(!p->attrib[0])
            {
                oldp->next='\0';
                break; //Конец списка
            }
            p->next=(struct attribute*)
            malloc(sizeof(at));
            if(!p->next){
                w_error1=draw_win(CTRWIN, CTRWIN, 45, 3, "Ошибка", popup, &errcolors);
                mprintf("Нет доступной памяти,нажмите любую клавишу");
                while(!kbhit());
                rmv_win(w_error1);clrscr();
                break;}
            oldp=p;
        }
    }
}
```

```
    p=p->next;
}
}
fclose(fp);
l_pos=t-1;
rmv_win(w_error);clrscr();
return;
}

//Функция восстановления базы знаний
clear_kbase()
{
    int t;
    struct attribute *p, *p2;

    for(t=0; t<l_pos; t++)
    {
        p=k_base[t].alist;
        while(p)
        {
            p2=p;
            free(p);
            p=p2->next;
        }
    }
    return;
}

//Сервисная функция
is_in(char ch, char *S)
{
    while(*S)
        if(ch==*S++) return 1;
    return 0;
}
```